Using CMMI to Balance Agile and Plan-driven Methods

Richard Turner
The George Washington University
OUSD(AT&L)/DS/SE

CMMI Technology Conference

Denver, CO
November 19, 2003

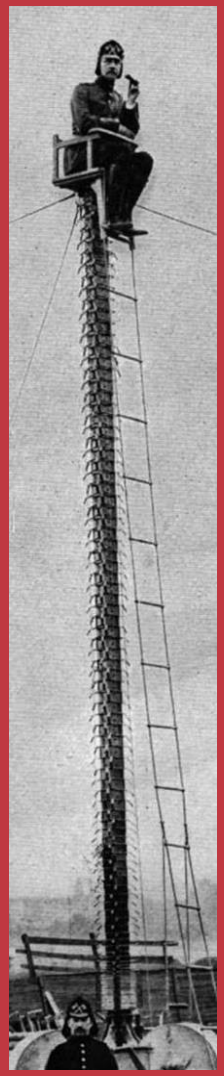
Background

- Success in any endeavor requires both agility and discipline
- Two approaches to software development
  - Plan-driven (SW-CMM, document-based, strong process)
  - Agile (XP, tacit knowledge, light process)
- Agile and plan-driven proponents are believers
- Both have strengths and ba[illegible] is needed

WARNING!
Generalizations Ahead…

Some Observations on Balancing

- Neither agile nor plan-driven methods provide a silver bullet
- Agile and plan-driven methods have home grounds where each clearly dominates
- Future developments will need both agility and discipline
- Some balanced methods are emerging

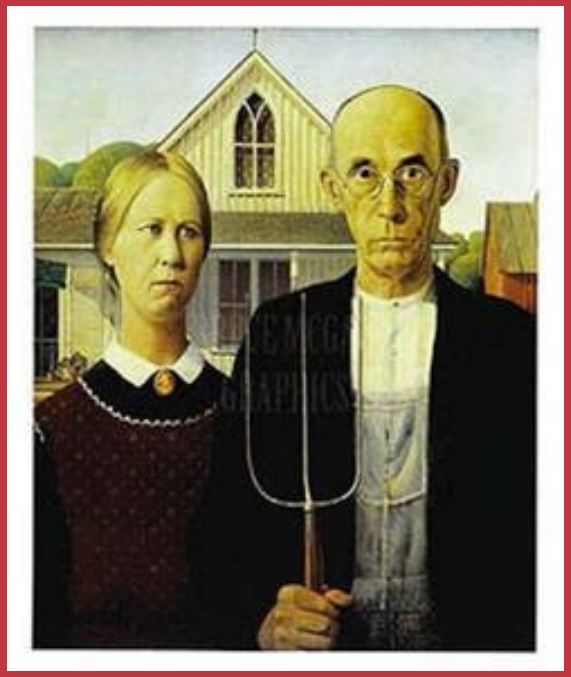


- It is better to build your method up than to tailor it down
- Methods are important, but potential silver bullets are more likely to be found in areas dealing with
  - People
  - Values
  - Communications
  - Expectations management

1. No Silver Bullet

- Brooks' werewolf concerns
  - Complexity, conformity, changeability, invisibility
- Agile methods
  - On target for changeability and invisibility
  - Miss on complexity and conformity
- Plan-driven methods
  - On target for conformity and invisibility
  - Miss on complexity and changeability
- Bullets can lose their efficacy as "wolves" evolve

2. Home Grounds Exist



- **Agile and plan-driven methods have definite home grounds**
  - **Environment where they are most likely to succeed**
  - **Extremes are rarely populated**
- **Five dimensions can help illustrate a project's or organization's home ground relationship**
  - **Size, Criticality, Dynamism, Personnel, Culture**

3. Future Applications Need Both

- **Historically**
  - **Many small, non-critical, well-skilled, agile culture, rapidly evolving projects**
  - **Many large, critical, mixed-skill, ordered culture, stable projects**
- **In the future**
  - **Large projects are no longer stable**
  - **Maintenance of extensive process and product plans will become too expensive**
  - **Complexity and conformity werewolves are waiting for agile projects**
  - **Attributes of both approaches will be needed**

4. Balanced Methods are Emerging

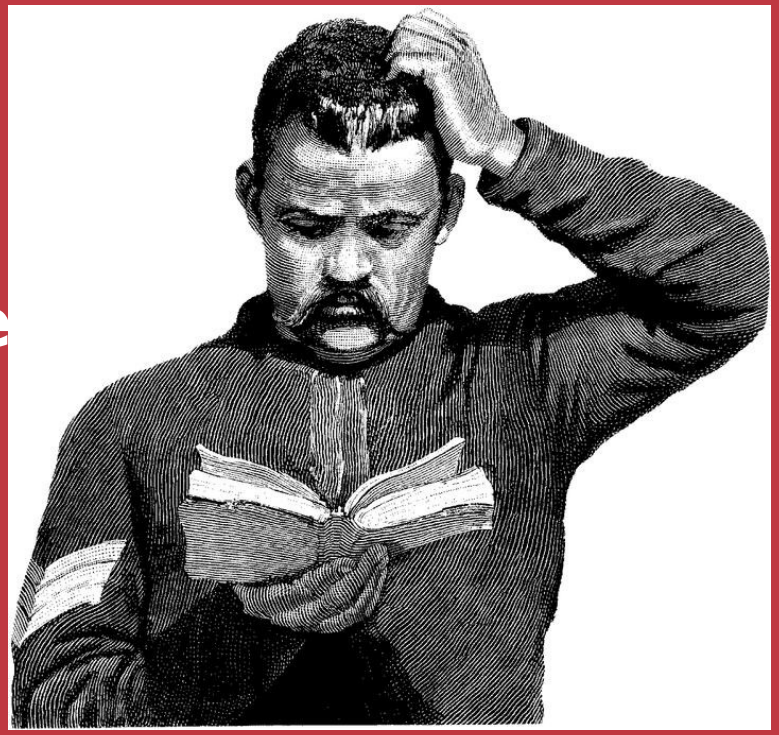
- Agile methods
  - Crystal Orange
  - DSDM
  - FDD
  - Lean Development
- Plan-Driven methods
  - Rational Unified Process
  - CMMI
- Hybrid
  - Boehm-Turner Risk-based
  - Manzo (AgileTek) Code Science/Agile Plus

5. Build up – Don’t Tailor Down

- Plan-driven methods traditionally
  - Have over defined processes
  - Advocate (or require) tailoring
  - Are rarely tailored well
- Agilists traditionally
  - Begin with the minimum
  - Add as needed (and justified by cost-benefit)
  - Have multiple core sets

6. Methods aren’t always the answer

- Agile movement has echoed a long line of warning calls
- Success of agile may be due as much to people factors as to technology
- Valuing people over processes is the most important factor in the agile manifesto



I know I saw something about that in the process somewhere…

People

- **Development is “of the people, by the people, for the people”**
- **Separation of concerns is increasingly harmful**

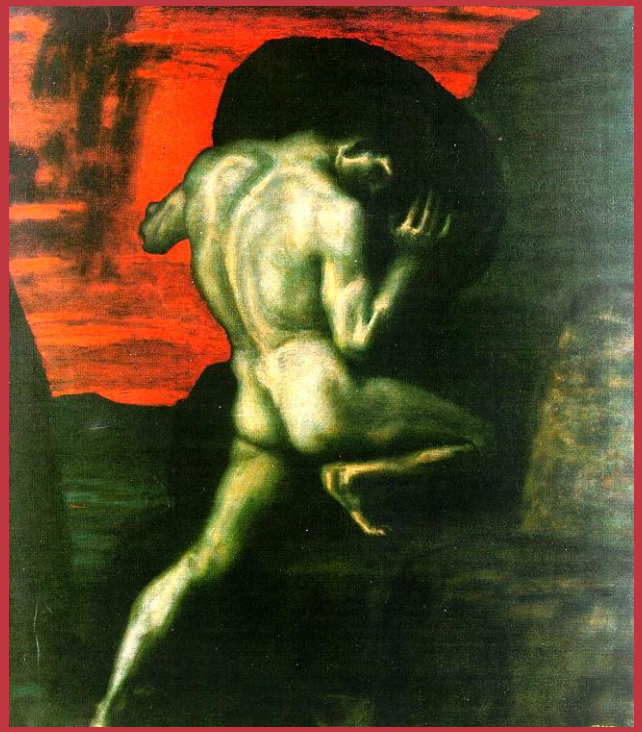
Values

- Reconciling values is a critical people-oriented task
- Stakeholders value different things
- Software engineering is usually value-neutral
- Process improvement and plan-driven methods are inwardly-focused
  - Aimed at productivity improvement
  - Not higher value to customer
- Agilists atte zation and negotiation

Communications

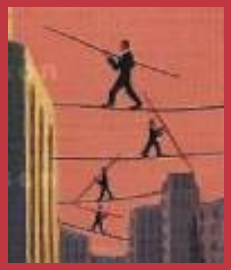
- Face it, most engineers can’t talk, much less write
- IKIWISI and management-by-rock reign
- Rapid change increases need for solid communication
- Few sources of guidance
  - Alistair Cockburn’s *Agile Software Development* is a good starting place

Expectations Management



Developers seem to like Sisyphean tasks…

- **Differences between successful and troubled projects is often expectations management**
- **SW developers have problems with EM**
  - **Strong desire to please**
  - **Avoid confrontation**
  - **Little confidence in prediction**
  - **Over confidence in abilities**
- **Most significant common factor in successful plan-driven/agile teams**
- **EM means not setting up unrealistic expectations**
  - **Process mastery**
  - **Preparation**
  - **Courage**

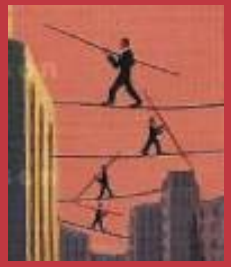


So… How does CMMI help balance?

- Flexibility
- Broader engineering and management scope
- Addresses people aspects
- High maturity

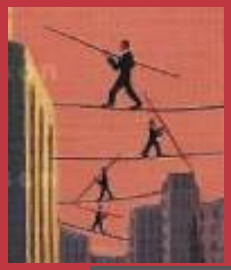
Flexibility

- Goals and practices are more flexible in CMMI
  - Need to span differences led to more general language
  - Alternative practices provide an entry point for innovative approaches
- Continuous representation provides more flexibility in deciding what PI should address



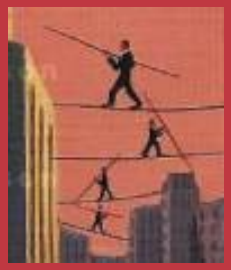
Broader Engineering and Management Scope

- Supports agile technical/management approaches
  - Product Integration PA can support continuous integration
  - Engineering and Support Pas (e.g. VAL, CM) are compatible with test-driven design and automated tools
- Agile methods are often both iterative and concurrent
  - Recursion of engineering PAs (e.g. RD, TS) supports iterative development
  - Broader scope allows multiple disciplines and approaches for different components
    - *Agile for emerging or rapidly evolving components*
    - *Plan-driven for well-understood or regulated components*



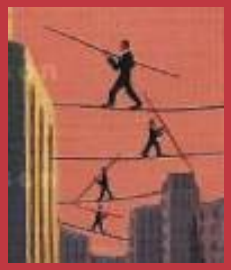
Addresses People Aspe

- Values
  - Effective SE (e.g. trade studies) cannot be value neutral (DAR)
- Communication
  - IPPD, shared vision, and stakeholder concerns make for more effective communications (IPM, PP)
- Expectation Management
  - Emphasis on measurement produces good data that enables effective expectation management (MA)

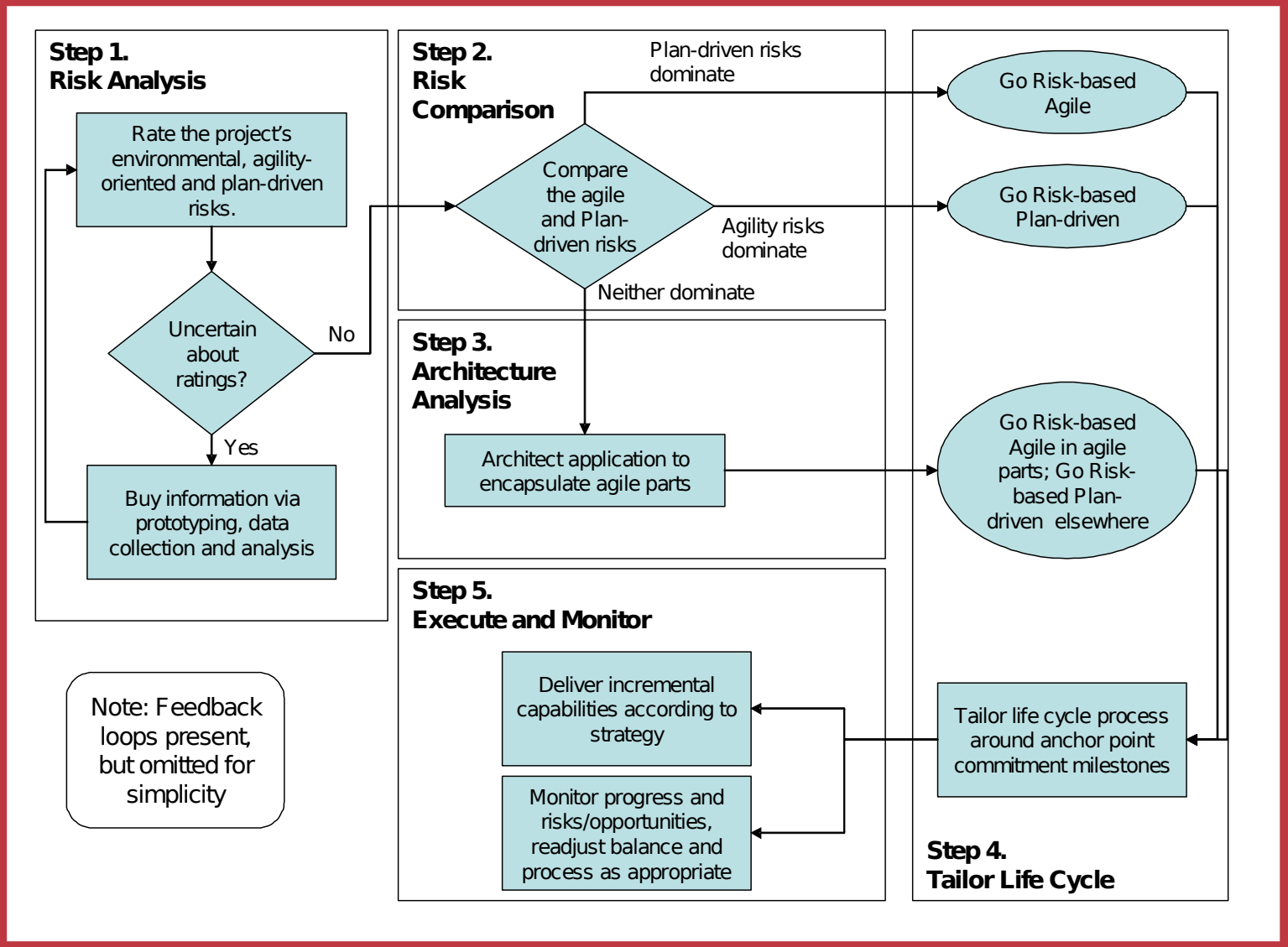


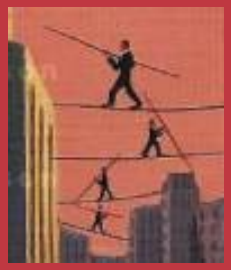
High Maturity

- **Innovation at level 5 argues for agile approaches**
- **Having both agile and plan-driven standard processes allows marketplace agility**
- **Application of Lean and Six Sigma techniques at high maturity levels eliminate [illegible] processes and results in [illegible] rather than tailor-do[illegible]**

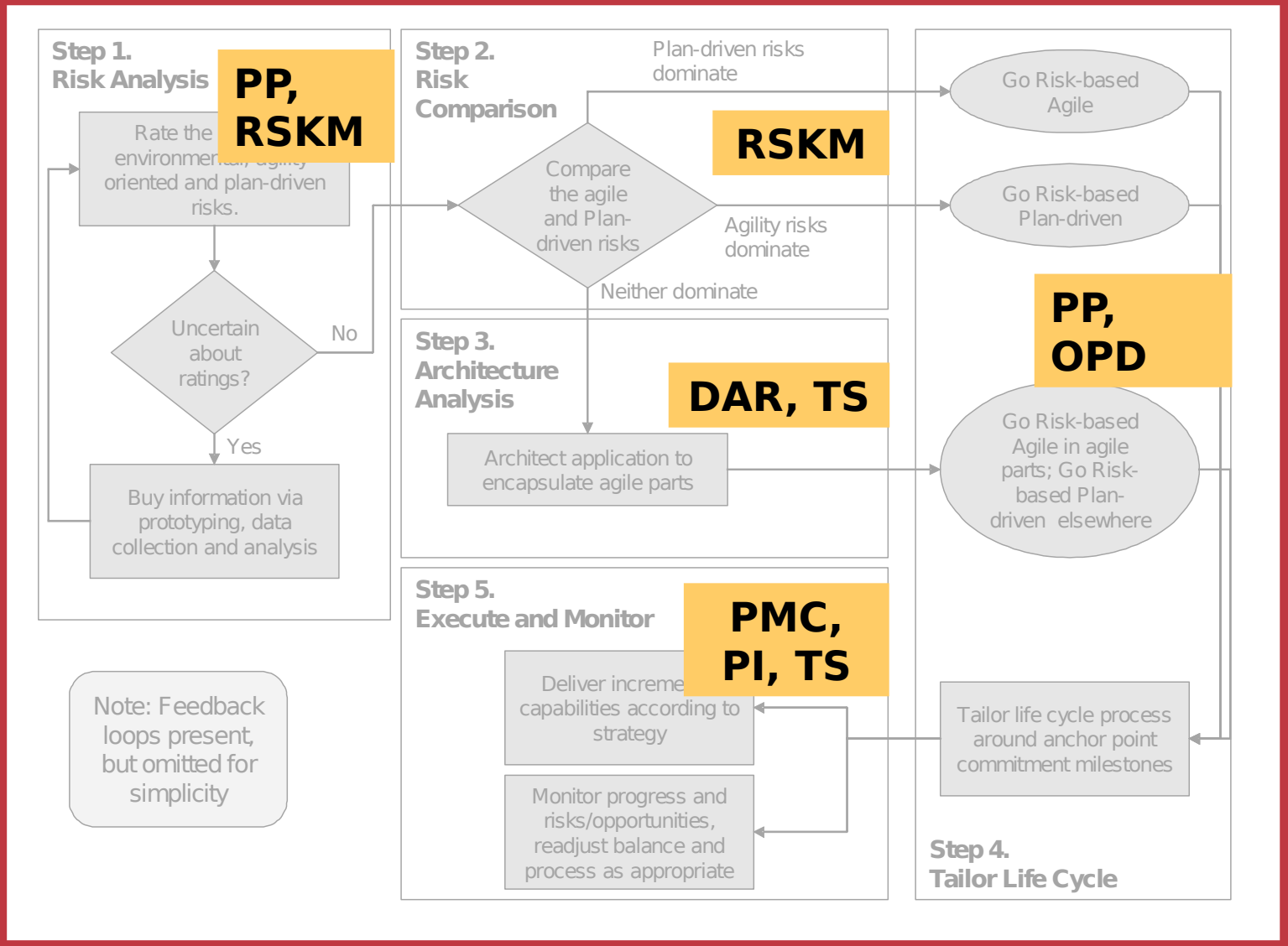


A Risk-based Balancing Process


Step 1.
Risk Analysis
Rate the project's environmental, agility-oriented and plan-driven risks.
Uncertain about ratings?
No
Yes
Buy information via prototyping, data collection and analysis
Step 2.
Risk Comparison
Plan-driven risks dominate
Compare the agile and Plan-driven risks
Agility risks dominate
Neither dominate
Step 3.
Architecture Analysis
Architect application to encapsulate agile parts
Step 5.
Execute and Monitor
Deliver incremental capabilities according to strategy
Monitor progress and risks/opportunities, readjust balance and process as appropriate
Note: Feedback loops present, but omitted for simplicity
Go Risk-based Agile
Go Risk-based Plan-driven
Go Risk-based Agile in agile parts; Go Risk-based Plan-driven elsewhere
Tailor life cycle process around anchor point commitment milestones
Step 4.
Tailor Life Cycle



How CMMI Supports the Process



Conclusions

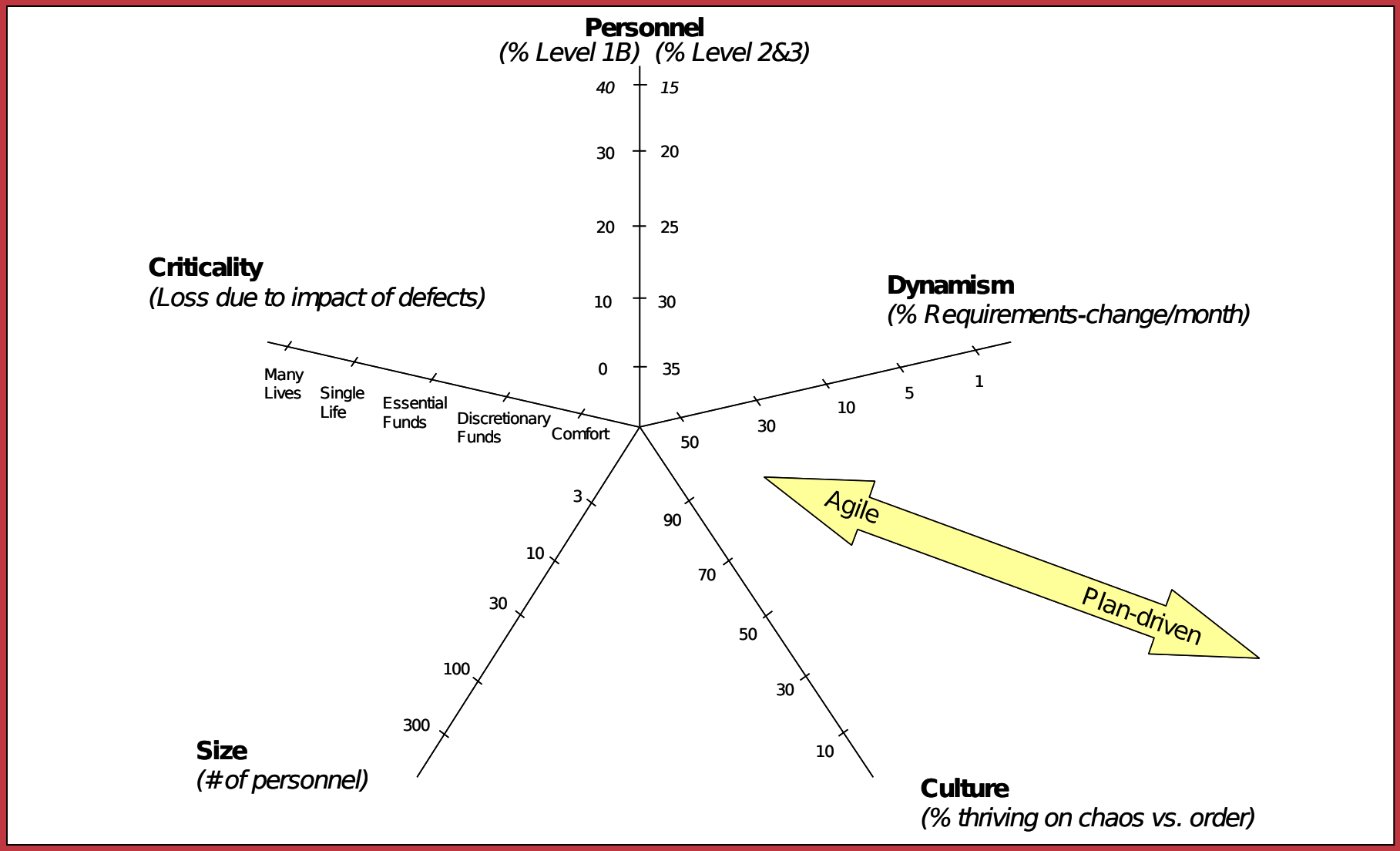
- Plan-driven and agile methods both aim to
  - Satisfy customers
  - Meet cost and schedule parameters
- Home grounds exist, but the opportunity for integration is expanding
- CMMI supports balancing methods
  - Flexible application of the model allows both plan-driven and agile methods
  - PA support to risk-based balancing process
- For more on the risk-based process, see
  - Boehm/Turner, *Balancing Agility and Discipline: A Guide for the Perplexed,* Addison Wesley, Boston, 2003.

Contact Information

Rich Turner
703.602.0851 x124
Rich.Turner.CTR@osd.mil

Back-up

Five Critical Dimensions



Summary of Home Grounds

Characteristics	Agile	Disciplined
Application		
Primary Goals	Rapid value; responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent; high change; project-focused	Stable; low-change; project/organization focused
Management		
Customer Relations	Dedicated on-site customers; focused on prioritized increments	As-needed customer interactions; focused on contract provisions
Planning/Control	Internalized plans; qualitative control	Documented plans, quantitative control
Communications	Tacit interpersonal knowledge	Explicit documented knowledge
Technical		
Requirements	Prioritized informal stories and test cases; undergoing unforseeable change	Formalized project, capability, interface, quality, forseeable evolution requirements
Development	Simple design; short increment; refactoring assumed inexpensive	Extensive design; longer increments; refactoring assumed expensive
Test	Executable test cases define requirements, testing	Documented test plans and procedures
Personnel		
Customers	Dedicated, collocated CRACK* performers	CRACK* performers, not always collocated
Developers	At least 30% full-time Cockburn level 2 and 3 experts; no Level 1B or -1	50% Cockburn Level 3s early; 10% throughout; 30% Level 0's workable; no